NBSIR 86-3417

User's Guide for SETKY-GETKY A Keyed Access System for the HP1000

Dorothy Bickham and David Neumann

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Chemical Physics
Chemical Thermodynamics Division
Gaithersburg, MD 20899

October 1986

Interim Report

U.S. DEPARTMENT OF COMMERCE
L BUREAU OF STANDARDS

NBSIR 86-3417

USER'S GUIDE FOR SETKY-GETKY A KEYED ACCESS SYSTEM FOR THE HP1000

Dorothy Bickham and David Neumann

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Chemical Physics
Chemical Thermodynamics Division
Gaithersburg, MD 20899

October 1986

Interim Report

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

User's Guide for SETKY-GETKY

A Keyed Access System for the HP1000

Documentation by
Dorothy Bickham

Programmed by
Dorothy Bickham
David Neumann

Program - September 1986

Preliminary

Copy

This is a preliminary copy of the manual for review and interim usage.

Table of Contents

INTRODUCTION — CHAPTER 1

SETKY-GETKY(1) is a keyed access system written for the Hewlett-Packard HP1000 mini-computer*. The basic characteristic of a keyed or indexed access system is that, in order to access data in a data file or data base, a separate file is created containing keys to the records in the data file and the locations where the records are stored. The file so created is called a key file or index. The key file is much smaller than the original data file and can be searched much more rapidly(2). Then the key file can be used to provide direct access to the data file. In the SETKY-GETKY system, the keys or keywords are defined by the user and serve to identify, for future retrieval, groups of records in the data file. The manual will describe just how this is accomplished.

SETKY-GETKY performs two major functions:

(1) Makes the help system on the computer more user-friendly by providing a quick and easy way for the users to obtain help information on requested topics.

(2) Provides rapid access to free formatted textual or tabular material stored in large data files on the computer.

To use the local default help system, the user only needs to enter

```
CI> GETKY, ?  or

CI> GETKY, help
```

to display a list of topics or keywords under which help information is available. To obtain help on a particular one of the keywords, enter

---

* Certain commercial equipment including computer software is identified in this paper in order to adequately specify procedures, experiments, and techniques used. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the materials, equipment, or software identified are necessarily the best available for the purpose.

CI> GETKY, <keyword>

In the following chapter, entitled GETKY, more detailed instructions are given on how to use the program GETKY to obtain help information stored on the HP1000. In addition, the more general use of GETKY to retrieve textual or tabular data from any indexed data file is described.

Chapter 3, SETKY, explains how a user may set up his own keyed access system. Briefly, to implement the SETKY-GETKY system, the user can start with an existing data file and then add to it certain SETKY commands to indicate what data should be accessed by what keyword. The keyword can be any ASCII character string, up to 24 characters in length, such as, author name, program name, abstract number, or chemical substance. The data can be textual or tabular information, a transfer command, or a command to run a program.

After the data file has been organized in this manner, the SETKY program is run to generate a key file containing all keywords in the data file and the locations of the data associated with each keyword. GETKY then uses this key file to produce a display of the data for any of the keywords. In its interactive mode, GETKY will also display on the terminal screen an alphabetized list of keywords that bracket a requested keyword if the keyword itself is not in the key file. The user may then select one of the displayed keywords or move forward or backward through a list of keywords. SETKY-GETKY also provides an automatic facility for transferring from one key file to another.

Chapters 4 and 5 describe some additional features of the SETKY-GETKY system. Chapter 4, TEXED[3], discusses a document processor compatible with SETKY-GETKY. A programmatic interface available between the SETKY-GETKY system and user-written application programs is outlined in Chapter 5.

Thus as the reader progresses through the manual, it should become apparent how this system can be useful to users with a wide range of computer experience.

(1) GETKY can be run by users possessing minimal computer skills to retrieve keyed data and display it, print it, or capture it in a file.

(2) SETKY can be run by users experienced in editing to set up keyed access to their own data files by inserting the appropriate SETKY commands.

(3) The programmatic interface available with SETKY-GETKY can be used by

programmers to access keyed data from within their own programs.

Chapter 6, Possible Enhancements, lists the enhancements currently under consideration. Any suggestions from users of the SETKY-GETKY System for additions or changes to the list would be appreciated.

GETKY	CHAPTER 2

The program GETKY searches the key file for a given key and then displays data from the data file for that key using the location retrieved from the key file.

To run GETKY, enter the following runstring:

```
CI>  GETKY[, key[, keyfile[, output[, [NI]]]]
```

where

key is the one-word ASCII character string that identifies the data to be displayed. For the special case when the key entered is "?" or "help" and the remaining parameters are omitted, a list of keywords will be displayed. If the key is omitted, the program will display an explanation of the program's usage and then halt.

keyfile is the file descriptor of the type 2 file containing the keys to the data file. It must have the form:

```
directory.../filename.key
or  directory.../filename.key::globaldir.
```

An example key file name is BIBLIO.KEY.
If omitted, keyfile defaults to HELP.KEY, the local default help key file (stored on the global directory, SFS, the Search File System directory).

output is the file or unit where the GETKY output is to be displayed or printed. The unit may be 1 for the terminal screen or the lu number of a printer. If omitted, output defaults to 1.

NI is used to specify non-interactive mode. If omitted, the default is interactive mode.

GETKY searches the key file for the given key. If the key is found, GETKY uses the location stored with the key to quickly access the data for that key in the data file.

If the data is textual or tabular information, then it is displayed on the requested output device. If output = 1, the data is displayed on the

screen. If a SETKY command that regulates the display is encountered within the data or if a "screenful of data" has already been displayed, then in the interactive mode the display pauses and the user is given the choice of displaying more data or aborting the display. In the non-interactive mode the display always continues. If output = the lu number of a printer, the program automatically spools and sends the data to the printer. SETKY commands that pause the display while going to the screen are modified to cause an advance to new page. If output = filename, the data is written to the named file and any SETKY commands for pausing the display are ignored. In the case where the file requested for output already exists, the user is given the choice of overwriting the file or aborting GETKY.

For example, to display on the screen all information indexed under the key, PRT3L, in the default help key file, HELP.KEY, enter

```
CI> GETKY, PRT3L
```

To send to the printer, lu=31, all data indexed under the key, AUTHORS, in the key file, BIBLIO.KEY, stored on the global directory, KEYS, enter

```
CI> GETKY, AUTHORS, BIBLIO.KEY::KEYS, 31
```

When the data under a key is a command to run a program, that program is scheduled to be run. Note that if the program is LI, to list a file, the output will go to the screen regardless of the value of output in the GETKY runstring, because LI allows no output choices. If the data under the key is a command to transfer to another key file, GETKY checks whether a new key is requested. If so, a search is made in the new key file for the new key; otherwise the search is made in the new key file for the current key.

If a requested key is not found in the specified key file and if the mode of GETKY is non-interactive, the program halts. If the mode is interactive, a list of keywords, that can be found in the key file and that bracket in alphabetical order the requested key, is displayed on the screen. The user is then provided with the choice of aborting GETKY, requesting a new key, or moving forward or backward in alphabetical order in the list of available keys. If, while the user is moving forward or backward through the list, he comes to the beginning or end of the list, one of two things will happen. If the data file contains a keyword for PRIORFILE or NEXTFILE and a new key file to transfer to, GETKY will automatically transfer to the requested key file and resume the search there for the keyword. If a backward move is requested and PRIORFILE is not defined or a forward move is requested and NEXTFILE is not defined, the program halts with a message.

It is possible to have the situation where the same keyword references more

than one set of data in the data file. If this occurs while GETKY is running in its non-interactive mode, all occurrences of the data for the keyword are displayed in succession. If this occurs while GETKY is running in its interactive mode, the display depends on the output device requested. If output is to be to a file, all occurrences are displayed in succession. If output is to a printer, all occurrences are displayed, each new one starting on a new page. If output is to the screen, the program pauses after displaying each occurrence and gives the user the option of displaying another occurrence or aborting GETKY.

For example, to write into the file, PRTHLP, all sets of data indexed under the key, PRINT, in the default indexed help file, enter

```
CI> GETKY, PRINT, HELP.KEY::SFS, PRTHLP    or

CI> GETKY, PRINT, ,PRTHLP
```

To display on the screen, all sets of data indexed under the key, DATACOMM, enter

```
CI> GETKY, DATACOMM
```

The program, GETKY, will display one set of data indexed under DATACOMM at a time, pausing after each to give the user the choice of aborting the program or displaying another set of data.

To display all sets of data, one after the other, without pausing, enter

```
CI> GETKY, DATACOMM, HELP.KEY::SFS, 1, NI       or

CI> GETKY, DATACOMM, , ,NI
```

SETKY	CHAPTER 3

3.1 Overview

The program SETKY generates a key file for use by its sister program, GETKY.

To run SETKY, enter the following runstring:

```
CI>  SETKY[, filedescr]
```

where

filedescr is the file descriptor of the ASCII data file to be keyed. It must must have the form:

```
directory.../filename.idx  or
directory.../filename.idx::globaldir.
```

An example data file name is BIBLIO.IDX. If omitted, the program will display a description of the program's usage and then halt.

The reason the data file is given a type extension of IDX and not, for example, DAT or TXT is so that it can easily be distinguished from other data files on the computer. It does contain data or textual material, but it also contains SETKY commands and thus is designed to be indexed.

Note that the file descriptor for the key file is set programmatically to be same as that of the data file, except that the type extension is changed from "idx" to "key". In the above example BIBLIO.KEY would be the key file generated by SETKY. IF A KEY FILE OF THE SAME NAME ALREADY EXISTS, IT WILL ALWAYS BE OVERWRITTEN!

The key file is a type 2 file which is used by GETKY. It contains a sorted list of all the keywords found and the record numbers and positions where their data can be located in the data file. SETKY also generates a list file whose file descriptor is set programmatically to be the same as that of the data file, except that the type extension is changed to "lst". In the above example, BIBLIO.LST would be the list file generated by SETKY. This

file is an ASCII file generated for purposes of documentation only and contains the names of all other key files, if any, that are the objects of transfer commands and the keywords under which the transfer commands are made. It also contains an alphabetized listing of all keywords processed by SETKY.

3.2 Data File

Basically the input data file for SETKY consists of a series of keywords, each followed by data. Any record with a double quote,", in column 1 is interpreted as a SETKY command by the SETKY-GETKY system. Any record with a period in column 1 is ignored except for those beginning with .INDEX or.PAGE which will be described shortly.

The ordering of the data file is as follows. A pair of double quotes, "", signifies that the next record contains a keyword. This next record must be of the form, "<keyword>, where keyword may be up to 24 characters in length. After the keyword comes the data record, or records, associated with it. Then another pair of double quotes leads into the next keyword and so on to the end of the file.

The data following a keyword may be in the form of text, a table, or a command. If text or a table, it must have an initial record of the form, "SS, and a trailing record of the form, "XX. Note that the leading double quote is required. Anywhere within the text or table, there may be a record or records of the form, "&. These records are used by GETKY to regulate the flow of the display of data to the screen or to a printer.

If a command follows a keyword, the command must be either a RU command to run a program or a TR command to transfer to another key file. For example, after the keyword, SKETCH, you might have the command, RU, LI, /HELP/SKETCH, which will list the sketch help file.

For SETKY-GETKY to be able to access data, the data MUST be preceded by at least one keyword. It is permissible to have multiple keywords access the same data. This may be achieved in either of two ways:

(1) List the keywords, each preceded by a double quote, in succession before the start of the data. Each of these keywords must be preceded by a record containing a pair of double quotes.

(2) List additional keywords within the data, i.e., between the "SS and "XX records, each keyword preceded by a double quote or, optionally,

by the string, .INDEX.

Note that the user can easily modify the data file by adding, changing, or deleting data and/or keywords. However, ANYTIME THE DATA FILE IS MODIFIED, SETKY MUST BE RERUN to update the key file in order that GETKY can access the data correctly.

Assuming that EDIT/1000[4] is used to edit the data file, there are certain size restrictions to consider. EDIT/1000 allows a record to be a maximum of 150 characters in length and the total number of records in the file is limited to approximately 32,000. In addition, as the SETKY program is currently written, the number of keyword entries in the data file can not exceed 1000.

3.3 SETKY Commands

Definitions of all available SETKY commands are as follows:

""
Signal that next record contains a keyword.

"<keyword>
<keyword> gives the ASCII representation for the keyword that is to be associated with the corresponding data. The keyword may be up to 24 characters in length. Note that the character string, KEYEND, is reserved for programmatic use by the SETKY-GETKY programs to indicate the end of the key file and is not available for use as a keyword.

.INDEX<keyword>
Optional method of giving the ASCII representation for a keyword. Note this method is only valid when used for additional keywords within the body of the data.

"SS
Signals start of data.

"XX
Signals end of data.

"NEXTFILE <keyfile>
Nextfile is a reserved keyword that is used to direct GETKY to transfer to the given keyfile. This transfer occurs when a user in the interactive mode enters a forward request in the list of keywords that is displayed on the

screen and there are no more succeeding keywords available in the current key file.

"PRIORFILE <keyfile>
Priorfile is a reserved keyword that is used to direct GETKY to transfer to the given keyfile. This transfer occurs when a user in the interactive mode enters a backward request in the list of keywords that is displayed on the screen and there are no more preceding keywords available in the current key file.

"&
May be placed within body of data, i.e., between "SS and "XX, to regulate the data display. If output is to the screen, in the interactive mode GETKY will stop the display and ask the user to indicate whether the display should continue or be aborted. If output is to a printer, an advance to the next page will be made. If output is to a file, "& is ignored.

.PAGE
Optional command that is equivalent to the "& command.

"RU <programname> [parameters]
Commands GETKY to run the program, programname, with the optional parameters as part of the runstring.

"TR <keyfile> [keyword]
Commands GETKY to transfer to the file, keyfile. If keyword is given, a search will be made for it in the key file transferred to. If keyword is omitted, a search will be made for the current keyword.

.[line of text]
If the beginning of the line of text is not equal to INDEX or PAGE, the entire line is treated as comments and not processed.

3.4 An Example

The following is an example of how to create and then access a data file for SETKY-GETKY. First create the following data file named PROGS.IDX:

```
Start of PROGS.IDX>>>
""
"BITOF
"SS
            BITOF -- Removing the 8th bit from characters

            11:46 AM  THU.,  9  OCT., 1986  Preliminary copy
```

BITOF is a program that turns off the 8th bit in every character in a data file if this bit is turned on. This is necessary sometimes for files retrieved by VTEP from remote computer sites, e.g., CAS Online. To run the program, enter

```
CI> BITOF
```

and then the program will prompt for the names of the input and output filenames.

DB 22 Nov 85

```
"XX
""
"SEND
"RU,LI,/HELP/SEND
""
"DI
"SS
```

Display Your Directory Tree

```
.INDEXDIRECTORY
```

Program DI lists the whole directory tree under the current (or any chosen) (global) directory and indicates the current working directory. The working directory can then be changed by positioning the cursor in the desired line.

Before stopping, DI makes a directory list of the new working directory.

Auto-linefeed must be off at the terminal (as is usual) .

```
.PAGE
```

To run enter:

```
CI> DI,directoryname
```

where directory name is either /globaldirname
or subdirname

if no directory name is given current WD is used.

Obtained from Antwerp Swap tape.

(DN/DB 28Apr85)

```
"XX
""
"VTEP
"TR DATACOMM.KEY
""
"manuals
"ss
```

Recommended Manuals for HP-1000

1. Getting Started With RTE-A $5.00
   HP
   manual part no. 92077-90039
2. RTE-A Quick Reference Guide $13.00
   HP
   manual part no. 92077-90020
3. Edit/1000 User's Guide $9.00
   HP
   manual part no. 92074-90001
4. FORTRAN 77 Reference Manual $25.00
   HP
   manual part no. 92836-90001
5. Grafit/1000 Users Manual $50.00
   Graphicus
   part no. G1000M
6. RTE-A Programmer's Reference Manual $20.00
   HP
   manual part no. 92077-90007

"&

7. Image/1000 II User Reference Manual $28.00
   HP
   manual part no. 92081-90001
8. Ask/1000 Reference Manual $30.00
   CCS
   stock no. S32300M
9. Math/1000 The Electronic Spreadsheet Manual $10.00
   COMPROG
   (no part no. needed)
10. Connect Reference Manual $25.00
    Interactive Computer Technology
    manual part no. 62100M
11. RTE-A Link User's Manual $7.00
    HP

```
            manual part no. 92077-90035
         12. Symbolic Debug/1000 User's Manual                    $10.00
            HP
            manual part no. 92860-90001
"XX
""
"plot
""
"graph
"ru grafit
""
"NEXTFILE
"TR,ERRMSG.KEY
""
"PRIORFILE
"tr bb.key
<<<End of PROGS.IDX
```

After the data file has been created, run SETKY as follows

```
      CI> SETKY, PROGS.IDX
```

SETKY will create a key file, PROGS.KEY and a list file, PROGS.LST.
PROGS.LST will contain the following information:

```
Start of PROGS.LST>>>
File PROGS.LST

Other keyfiles requested for transfer -- Keywords making the request:

DATACOMM.KEY -- VTEP
ERRMGS.KEY -- NEXTFILE
bb.key -- PRIORFILE

Keywords found:

BITOF
DI
DIRECTORY
GRAPH
MANUALS
NEXTFILE
PLOT
PRIORFILE
SEND
```

```
VTEP

 Total number of keywords found:  10
<<<End of PROGS.LST
```

Then the user is ready to run GETKY to access data from PROGS.IDX.

1. To display on the screen all information indexed under the key, BITOF, enter

```
CI> GETKY, BITOF, PROGS.KEY
```

2. To display on the screen all information indexed under the key, DI, enter

```
CI> GETKY, DI, PROGS.KEY
```

   All data is displayed up to the .PAGE record. Then the display pauses. The user may chose to continue the display or abort it.

3. To display all data keyed under DI without pausing at the .PAGE, enter

```
CI> GETKY, DI, PROGS.KEY, 1, NI  or

CI> GETKY, DI, PROGS.KEY, , NI
```

   Note that the key, DIRECTORY, could be substituted for the key, DI. This is because DIRECTORY is another keyword that accesses the same data as DI through the use of the .INDEX SETKY command.

4. To produce on printer, lu=6, the list of manuals indexed under the keyword, MANUALS, enter

```
CI> GETKY, MANUALS, PROGS.KEY, 6
```

   The printout will be two pages long, divided at the "& record in PROGS.IDX.

5. To list the file, /HELP/SEND, enter

```
CI> GETKY, SEND, PROGS.KEY
```

   and this will cause LI/HELP/SEND to be executed, because the data indexed under SEND is a "RU SETKY command.

6. To run the program, GRAFIT, enter

```
CI> GETKY, GRAPH, PROGS.KEY   or

CI> GETKY, PLOT, PROGS.KEY
```

7. To retrieve data keyed under VTEP, which is located in DATACOMM.IDX and keyed in DATACOMM.KEY, and store it in the file, VTEPFILE, enter

```
CI> GETKY, VTEP, PROGS.KEY, VTEPFILE
```

8. If a key is given in the GETKY runstring which is not in the key file and GETKY is running in the interactive mode, i.e., the NI parameter is not given in the runstring, a list of keys is displayed. For example, enter

```
CI> GETKY, AUTHORS, PROGS.KEY
```

AUTHORS is not keyed in PROGS.KEY so a list of valid keywords will be displayed. If the user wants to see more keywords, he can move forward or backward through the list of keywords. Because this example is for a data file with so few keys (only 10), moving backward immediately comes to the beginning of the list. In this example, PRIORFILE has been defined so a transfer to the key file, BB.KEY results. If AUTHORS is a valid keyword in BB.KEY, the data will be displayed; otherwise another list of keywords, this time from BB.KEY, will be displayed.

TEXED	CHAPTER 4

Of course, one of the major advantages of a keyed access system is that it provides very fast access to data in large files. An additional advantage of the SETKY-GETKY system is its compatibility with TEXED, the document processor for the HP1000. TEXED was adapted by Bruce H. Stowell, Jim Bridges, Bill Hassell, and John Johnson from earlier versions of document processors. It is a public domain program and is part of the contributed software library available from INTEREX(3).

With only one data file, it is possible to provide keyed access using SETKY-GETKY and also generate a text-edited document using TEXED. This is possible because all the commands for SETKY-GETKY begin with a double quote in column 1, and TEXED treats as comments records beginning with a double quote in column 1. On the other hand, all TEXED commands begin with a period in column 1, and SETKY-GETKY ignores all records beginning with a period in column 1 with two exceptions, .INDEX and .PAGE, which are interpreted in an analogous manner by both programs.

An example of a situation where the compatibility between SETKY-GETKY and TEXED would be very useful is the development of a computerized "help" facility. The user could construct one data file with help instructions pertaining to a variety of commands or programs. Then by adding the appropiate SETKY commands, and then running SETKY, he will have a key file. GETKY can reference this key file to display information on any keyed command or program described in the data file. The display output from GETKY can be directed to the screen, to a printer, or into a file. In addition, by inserting the necessary TEXED commands into the data file and then processing it with TEXED, a document or manual can be produced which contains all the textual information on the commands and programs described in the data file.

PROGRAMMATIC INTERFACE	CHAPTER 5

5.1 Method 1 RuSetKy and RuGetKy Subroutines

It is possible for users to access, from within their own programs, data that has been keyed by SETKY. One way to accomplish this is to schedule the program, GETKY, to be run from their programs. The subroutine, RuGetKy, will schedule GETKY for the user. If the data has not yet been keyed, the subroutine, RuSetKy, may be called first to schedule SETKY.

(1) RuSetKy is a subroutine that schedules the program, SETKY, to run with wait. Usage is

```
Call RuSetKy (datafile, error)
```

where

datafile is the filedescriptor of the data file to be keyed.

error is returned as an integer value.
0 if SETKY was scheduled.
-1 otherwise.

Note: Calling RuSetKy to run SETKY will result in the creation of two files:

(1) A key file that has the same file descriptor as datafile except that the type extension is "key".

(2) A list file that has the same file descriptor as datafile except that the type extension is "lst".

(2) RuGetKy is a subroutine that schedules the program, GETKY, to run with wait. Usage is

```
Call RuGetKy (key, keyfile, output, parm, error)
```

where

key is the one-word ASCII character string that identifies the data to be retrieved or displayed.

keyfile is the filedescriptor of the key file where key has been indexed.

output is the file or unit where the GETKY output is to be sent. If output is the name of a file, then the data retrieved by GETKY, after it has been scheduled, is written to this file. Then, to access the data, the user can read from this file following the RuGetKy call. It is also permissible for output to be 1 for the terminal screen or the lu number of a printer.

parm is a two-character string used to specify mode:
NI for non-interactive.
ni also acceptable for non-interactive.
-- anything else will default to interactive.

error is returned as an integer value:
0 if GETKY was scheduled.
-1 otherwise.

Note: In the special case where the data indexed under the key given in the RuGetKy parameter list is not textual or tabular material, but rather a RU command, then the RU command itself is executed. That is, the Subroutine RuGetKy schedules GETKY, which, in turn, schedules the program requested in the RU command.

5.2 Method 2 Special Purpose Subroutines

Another way for users to access, from within their own programs, data that has been keyed by SETKY is through the use of subroutines that have been written especially for this purpose.

(1) OpenKy is a subroutine that opens a data file and its associated key file. Usage is

Call OpenKy (datafile, error)

where

datafile is the name of the data file to be opened. For example, if datafile is the file, BIBLIO.IDX, OpenKy will open BIBLIO.IDX and also the key file, BIBLIO.KEY.

error is returned as an integer value:
0 if the opens were successful.
-1 otherwise.

(2) PositionKy is a subroutine that positions a previously opened data file to the place where the data indexed under a given key begins. Usage is

```
Call PositionKy (key, datafile, mode, error)
```

where

key is the keyword under which the desired data has been indexed.

datafile is the name of the data file containing the indexed data.

mode is an integer that specifies the type of search to be made for the key:
0 to rewind key file before search for key.
1 to search from current position of key file.
Note: Use mode = 1 to locate successive occurrences of a duplicate key. Mode = 0 will ALWAYS position the data file to the location where data is stored for the first occurrence of the requested key.

error is returned as an integer value:
0 if the positioning was successful and data was found.
1 if the positioning was successful and a TR command was found.
2 if the positioning was successful and a RU command was found.
-1 otherwise.

Note: After the data file has been positioned, the user may proceed as follows, depending on the error value returned.

If error = 0 to retrieve the data, see Subroutine ReadKy below.
1 to perform the transfer and then to continue the search for data in the new data file, see Subroutine TransferKy below.
2 to execute the run command, see Subroutine RuGetKy in the previous section.

(3) ReadKy is a subroutine that reads one record from a previously opened and positioned data file. Usage is

```
Call ReadKy (datafile, cbuf, length, mode, error)
```

where

datafile is the name of the data file to be read.

cbuf is the character string buffer in which the data read is to be returned.

length is the integer length of cbuf, i.e., LEN(cbuf).

mode is an integer that determines the type of read to be performed. Set mode equal to
0 To read data only, i.e., any SETKY or TEXED commands encountered in the data file will not be returned.
1 To read data and TEXED commands. SETKY commands will not be returned.

error is returned as an integer value:
0 if the data has been successfully retreived.
-1 if end-of-data SETKY command was found.
-2 if length of cbuf was too small to hold all the data (in which case the first length characters of the data read are returned.)
-3 if read was unsuccessful.

(4) TransferKy is a subroutine that executes a transfer command to a new data file and then positions the new data file to the place where the data begins for the requested key. Usage is

```
Call TransferKy (key, datafile, error)
```

where

key is the current key.
If the transfer command requests a new key, key is returned equal to the newly requested key; else key is returned unchanged.

datafile is the current data file that contains a transfer command indexed under key. If the transfer is successful, datafile is returned equal to the new data file corresponding to the key file requested in the transfer command.

error is returned as an integer value:
0 if the transfer and positioning were successful and data was found.
1 if the transfer and positioning were successful and another TR command was found.
2 if the transfer and positioning were successful and a RU command was found.
-1 otherwise.

Note: TransferKy automatically closes the currently opened key and data files and opens the new ones.

(5) CloseKy is a subroutine that closes a data file and its associated key file. Usage is

```
Call CloseKy (datafile, error)
```

where

datafile is the name of the data file to be closed. For example, if datafile is the file, BIBLIO.IDX, CloseKy will close BIBLIO.IDX and also the key file, BIBLIO.KEY.

error is returned as an integer value:
0 if the closes were successful.
-1 otherwise.

POSSIBLE ENHANCEMENTS	CHAPTER 6

1. Improvements to the list file generated by SETKY:
   (a) Include a list in alphabetical order of all programs that are requested to be run by data under keys in data file with the requesting keys.
   (b) Alphabetize the list of key files requested for transfer before they are output.
   (c) When the list of keywords found is output, follow each keyword by the number of times that it occurred in the data file. (List currently repeats duplicate keywords.)

2. In GETKY, allow an optional match on partial keyword. For example, if LOC is the requested key, it could match on LOCATION.

3. In GETKY, add the ability to generate a "select" file of all the different locations of data for a key that occurs more than once in a key file. Then incorporate the option to display all data from either the union or intersection of two or more of these select files. For example, the user could build a select file for the key, calcium, and another select file for the key, oxygen. Then, by chosing to display the intersection of the two select files, the display generated would be restricted to data that has been keyed BOTH on calcium and on oxygen.

4. Ability with a subroutine to run TEXED on a file of data previously retrieved by the calling program via the subroutines in method 2 of the programmatic interface.

5. Add the option of storing with each key a short ASCII description of the key. Then when GETKY displays on the screen the list of keys, it can also display the description of each key.

6. Allow the option through a new SETKY command of having data from other files merged with the data in the data file when GETKY retrieves data. (This would be similar to the TEXED .ADD command.)

7. Speed up the program's execution time:
   (a) A faster method of sorting the keys in SETKY could possibly be used.

(b) The search routine in GETKY for finding the requested key in the key file could be optimized.

REFERENCES

(1) "SETKY-GETKY Key-Accessed Text System" is included in the CSL/1000 release for 1986 (release 2625) from INTEREX, 680 Almanor Avenue, Sunnyvale, CA 94086, and is available from the authors upon request.

(2) Atre, S., *Data Base: Structured Techniques for Design, Performance, and Management*. John Wiley & Sons, Inc., New York, 1980. Page 197.

(3) "TEXED User's Guide, A Document Processor for the HP1000," documentation edited by Bill Hassell, adapted by Bruce H. Stowell, Jim Bridges, Bill Hassell, and John Johnson. The TEXED program and manual are included in the CSL/1000 library available from INTEREX, 680 Almanor Avenue, Sunnyvale, CA 94086.

(4) EDIT/1000 is a Hewlett-Packard product and is described in the manual *EDIT/1000 User's Guide* (August 1980), Part No. 92074-90001, Hewlett-Packard Co., 11000 Wolfe Road, Cupertino, CA 95014.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 86-3417	2. Performing Organ. Report No.	3. Publication Date OCTOBER 1986

4. TITLE AND SUBTITLE

User's Guide for SETKY-GETKY
A Keyed Access System for the HP1000

5. AUTHOR(S)

Dorothy Bickham and David Neumann

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Interim

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

SETKY-GETKY is a keyed access system written for the HP1000 mini-computer. It performs two major functions: (1) makes the help system on the computer more user-friendly by providing a quick and easy way for the users to obtain help information on requested topics and (2) provides keyed access to free formatted textual or tabular material. Some of the special features available with this system include the ability to transfer automatically between key files and the option of inserting run commands in the indexed data files.

The SETKY-GETKY system is completely compatible with TEXED, the document processor available on the CSL Library distributed by INTEREX, The International Association of Hewlett-Packard Computer Users. This has the advantage that a user can produce a keyed online help system for the HP1000 and also a text-edited user's guide from the same data file by inserting both SETKY commands and TEXED commands in the file.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Help system; HP1000 mini-computer; indexed; key file; keyed access; keys; TEXED document processor

13. AVAILABILITY

☒ Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

30

15. Price

$9.95

USCOMM-DC 6043-P80